FRONT
not provided

js:Sonar way

2020-04-01

目录

1. FRONT

报告提供了项目指标的概要，显示了与项目质量相关的最重要的指标。如果需要获取更详细的信息，请登陆网站进一步查询。

报告的项目为FRONT，生成时间为2020-04-01，使用的质量配置为 js:Sonar way，共计 89条规则。

1.1. 概述

编码问题

Bug 3	可靠性修复工作 25min
漏洞 0	安全修复工作 0min
坏味道 23	技术债务 3h48min

26 问题	开启问题	26
	重开问题	0
	确认问题	0
	误判问题	0
	不修复的问题	0
	已解决的问题	0
	已删除的问题	0
	阻断	1
	严重	1
	主要	15
	次要	9
	提示	0

静态分析

项目规模

3710 代码行数	行数	5964
	方法	389
	类	0
	文件	40
	目录	30
	重复行(%)	5.1

复杂度

878 复杂度	类	N/A
	方法	2.2
	文件	22.0

注释(%)

19.9 注释(%)	注释行数	921

1.2. 问题分析

违反最多的规则TOP10	
Dead stores should be removed	8
Unused local variables and functions should be removed	6
Variables and functions should not be redeclared	3
Comma operator should not be used	1
A "while" loop should be used instead of a "for" loop	1
Function calls should not pass extra arguments	1
Jump statements should not be followed by dead code	1
Variables should be declared explicitly	1
Collection and array contents should be used	1
Unnecessary imports should be removed	1

违规最多的文件TOP5	
history.js	4
qrcode.js	4
search.js	4
event.js	3
about.js	2

复杂度最高的文件TOP5	
qrcode.js	192
es6-promise.js	158
event.js	139
search.js	51
utils.js	41

重复行最多的文件TOP5	
search.js	69
index.js	62
resultItem.js	60
about.js	58
history.js	57

1.3. 问题详情

规则	Dead stores should be removed

规则描述	A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value
	only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources.
	Therefore all calculated values should be used.
	Noncompliant Code Example
	
	i = a + b; // Noncompliant; calculation result not used before value is overwritten
	i = compute();
	
	Compliant Solution
	
	i = a + b;
	i += compute();
	
	Exceptions
	This rule ignores initializations to -1, 0, 1, null , undefined , true , false , "" ,
	[] and {} .
	This rule also ignores variables declared with object destructuring using rest syntax (used to exclude some properties from object):
	
	let {a, b, ...rest} = obj; // 'a' and 'b' are ok
	doSomething(rest);
	
	let [x1, x2, x3] = arr; // but 'x1' is noncompliant, as omitting syntax can be used: "let [, x2, x3] = arr;"
	doSomething(x2, x3);
	
	See
	
	MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')
	CERT, MSC13-C. - Detect and remove unused values
	CERT, MSC56-J. - Detect and remove superfluous code and values

文件名称	违规行
history.js	44, 122
qrcode.js	717
search.js	54
event.js	231
about.js	42
index.js	188
infoUpload.js	205

规则	Unused local variables and functions should be removed

规则描述	If a local variable or a local function is declared but not used, it is dead code and should be removed. Doing so will improve maintainability
	because developers will not wonder what the variable or function is used for.
	Noncompliant Code Example
	
	function numberOfMinutes(hours) {
	var seconds = 0; // seconds is never used
	return hours * 60;
	}
	
	Compliant Solution
	
	function numberOfMinutes(hours) {
	return hours * 60;
	}

文件名称	违规行
history.js	44, 122
search.js	54
about.js	42
index.js	188
infoUpload.js	205

规则	Variables and functions should not be redeclared

规则描述	

	This rule checks that a declaration doesn't use a name that is already in use. Indeed, it is possible to use the same symbol multiple times as
	either a variable or a function, but doing so is likely to confuse maintainers. Further it's possible that such reassignments are made in error, with
	the developer not realizing that the value of the variable is overwritten by the new assignment.
	This rule also applies to function parameters.
	Noncompliant Code Example
	
	var a = 'foo';
	function a() {} // Noncompliant
	console.log(a); // prints "foo"
	
	function myFunc(arg) {
	  var arg = "event"; // Noncompliant, argument value is lost
	}
	
	fun(); // prints "bar"
	
	function fun() {
	  console.log("foo");
	}
	
	fun(); // prints "bar"
	
	function fun() { // Noncompliant
	  console.log("bar");
	}
	
	fun(); // prints "bar"
	
	Compliant Solution
	
	var a = 'foo';
	function otherName() {}
	console.log(a);
	
	function myFunc(arg) {
	  var newName = "event";
	}
	
	fun(); // prints "foo"
	
	function fun() {
	  print("foo");
	}
	
	fun(); // prints "foo"
	
	function printBar() {
	  print("bar");
	}
	
	printBar(); // prints "bar"

文件名称	违规行
search.js	96
resultItem.js	77, 55

规则	Comma operator should not be used

规则描述	The comma operator takes two expressions, executes them from left to right and returns the result of the second one. Use of this operator is
	generally detrimental to the readability and reliability of code, and the same effect can be achieved by other means.
	Noncompliant Code Example
	
	i = a += 2, a + b; // What's the value of i ?
	
	Compliant Solution
	
	a += 2;
	i = a + b;
	
	Exceptions
	Use of comma operator is tolerated in initialization and increment expressions of for loops.
	
	for(i = 0, j = 5; i < 6; i++, j++) { ... }
	
	See
	
	MISRA C:2004, 12.10 - The comma operator shall not be used.
	MISRA C++:2008, 5-18-1 - The comma operator shall not be used.
	MISRA C:2012, 12.3 - The comma operator should not be used

文件名称	违规行
qrcode.js	339

规则	A "while" loop should be used instead of a "for" loop

规则描述	When only the condition expression is defined in a for loop, and the initialization and increment expressions are missing, a
	while loop should be used instead to increase readability.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	for (;condition;) { /*...*/ }
	
	Compliant Solution
	
	while (condition) { /*...*/ }

文件名称	违规行
qrcode.js	414

规则	Function calls should not pass extra arguments

规则描述	You can easily call a JavaScript function with more arguments than the function needs, but the extra arguments will be just ignored by function execution.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	function say(a, b) {
	print(a + " " + b);
	}
	
	say("hello", "world", "!"); // Noncompliant; last argument is not used
	
	Exceptions
	No issue is reported when arguments is used in the body of the function being called.
	
	function doSomething(a, b) {
	compute(arguments);
	}
	
	doSomething(1, 2, 3) // Compliant
	
	See
	
	MISRA C:2004, 16.6 - The number of arguments passed to a function shall match the number of parameters.
	MITRE, CWE-628 - Function Call with Incorrectly Specified Arguments
	CERT, DCL07-C. - Include the appropriate type information in function declarators
	CERT, EXP37-C. - Call functions with the correct number and type of arguments

文件名称	违规行
es6-promise.js	269

规则	Jump statements should not be followed by dead code

规则描述	

	Jump statements (return , break and continue) and throw expressions move control flow out of the current code block. So any statements that come after a jump are dead code.
	Noncompliant Code Example
	
	function fun(a) {
	var i = 10;
	return i + a;
	i++; // Noncompliant; this is never executed
	}
	
	Compliant Solution
	
	function fun(int a) {
	var i = 10;
	return i + a;
	}
	
	Exceptions
	This rule ignores unreachable break statements in switch clauses.
	
	switch (x) {
	case 42:
	return 43;
	break; // Compliant
	default:
	doSomething();
	}
	
	Hoisted variables declarations without initialization are always considered reachable.
	
	function bar() {
	return x = function() {
	x.foo = 42;
	}
	var x;
	}
	
	See
	
	MISRA C:2004, 14.1 - There shall be no unreachable code
	MISRA C++:2008, 0-1-1 - A project shall not contain unreachable code
	MISRA C++:2008, 0-1-9 - There shall be no dead code
	MISRA C:2012, 2.1 - A project shall not contain unreachable code
	MISRA C:2012, 2.2 - There shall be no dead code
	MITRE, CWE-561 - Dead Code
	CERT, MSC56-J. - Detect and remove superfluous code and values
	CERT, MSC12-C. - Detect and remove code that has no effect or is never executed

文件名称	违规行
event.js	168

规则	Variables should be declared explicitly

规则描述	JavaScript variable scope can be particularly difficult to understand and get right. The situation gets even worse when you consider the accidental creation of global variables, which is what happens when you declare a variable inside a function or the for clause of a for-loop without using the let , const or var keywords. let and const were introduced in ECMAScript 2015, and are now the preferred keywords for variable declaration.
	Noncompliant Code Example
	
	function f(){
	i = 1; // Noncompliant; i is global
	
	for (j = 0; j < array.length; j++) { // Noncompliant; j is global now too
	// ...
	}
	}
	
	Compliant Solution
	
	function f(){
	var i = 1;
	
	for (let j = 0; j < array.length; j++) {
	// ...
	}
	}

文件名称	违规行
qrcode.js	665

规则	Collection and array contents should be used

规则描述	When a collection is populated but its contents are never used, then it is surely some kind of mistake. Either refactoring has rendered the collection moot, or an access is missing.
	This rule raises an issue when no methods are called on a collection other than those that add or remove values.
	Noncompliant Code Example
	
	function getLength(a, b, c) {
	  const strings = []; // Noncompliant
	  strings.push(a);
	  strings.push(b);
	  strings.push(c);
	
	  return a.length + b.length + c.length;
	}
	
	Compliant Solution
	
	function getLength(a, b, c) {
	  return a.length + b.length + c.length;
	}

文件名称	违规行
event.js	512

规则	Unnecessary imports should be removed

规则描述	There's no reason to import modules you don't use; and every reason not to: doing so needlessly increases the load.
	Finally, importing a module twice is pointless and confusing.
	Noncompliant Code Example
	
	import A from 'a'; // Noncompliant, A isn't used
	import { B1 } from 'b';
	
	console.log("My first JavaScript...");
	
	import { B1 } from 'b'; // Noncompliant, already imported
	
	console.log(B1);
	
	Compliant Solution
	
	import { B1 } from 'b';
	
	console.log("My first JavaScript...");
	
	console.log(B1);

文件名称	违规行
FRONT:app.js	2

规则	Properties of variables with "null" or "undefined" values should not be accessed

规则描述	When a variable is assigned an undefined or null value, it has no properties. Trying to access properties of such a variable anyway results in a TypeError , causing abrupt termination of the script if the error is not caught in a catch block. But instead of catch -ing this condition, it is best to avoid it altogether.
	Noncompliant Code Example
	
	if (x === undefined) {
	console.log(x.length); // Noncompliant; TypeError will be thrown
	}
	
	See
	
	MITRE, CWE-476 - NULL Pointer Dereference
	CERT, EXP34-C. - Do not dereference null pointers
	CERT, EXP01-J. - Do not use a null in a case where an object is required

文件名称	违规行
search.js	96

规则	Extra semicolons should be removed

规则描述	Extra semicolons (;) are usually introduced by mistake, for example because:
	
	It was meant to be replaced by an actual statement, but this was forgotten.
	There was a typo which lead the semicolon to be doubled, i.e. ;;
	.
	There was a misunderstanding about where semicolons are required or useful.
	
	Noncompliant Code Example
	
	var x = 1;; // Noncompliant
	
	function foo() {
	}; // Noncompliant
	
	Compliant Solution
	
	var x = 1;
	
	function foo() {
	}
	
	See
	
	MISRA C:2004, 14.3 - Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment provided that
	the first character following the null statement is a white-space character.
	MISRA C++:2008, 6-2-3 - Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment, provided
	that the first character following the null statement is a white-space character.
	CERT, MSC12-C. - Detect and remove code that has no effect or is never
	executed
	CERT, MSC51-J. - Do not place a semicolon immediately following an if, for,
	or while condition
	CERT, EXP15-C. - Do not place a semicolon on the same line as an if, for,
	or while statement

文件名称	违规行
es6-promise.js	517

1.4. 质量配置

质量配置	js:Sonar way Bug:41 漏洞:5 坏味道:43	

规则	类型	违规级别
Callbacks of array methods should have return statements	Bug	阻断
Loops should not be infinite	Bug	阻断

"yield" expressions should not be used outside generators	Bug	阻断
"in" should not be used with primitive types	Bug	严重
Function calls should not pass extra arguments	Bug	严重
"Symbol" should not be used as a constructor	Bug	严重
Results of "in" and "instanceof" should be negated rather than operands	Bug	严重
"super()" should be invoked appropriately	Bug	严重
Destructuring patterns should not be empty	Bug	主要
Conditionally executed blocks should be reachable	Bug	主要
Jump statements should not occur in "finally" blocks	Bug	主要
Property names should not be duplicated within a class or object literal	Bug	主要
"NaN" should not be used in comparisons	Bug	主要
Return values from functions without side effects should not be ignored	Bug	主要
Generators should "yield" something	Bug	主要
Function argument names should be unique	Bug	主要
Related "if/else if" statements and "cases" in a "switch" should not have the same condition	Bug	主要
All branches in a conditional structure should not have exactly the same implementation	Bug	主要
The output of functions that don't return anything should not be used	Bug	主要
Values should not be uselessly incremented	Bug	主要
Jump statements should not be followed by dead code	Bug	主要
Special identifiers should not be bound or assigned	Bug	主要
Properties of variables with "null" or "undefined" values should not be accessed	Bug	主要
A "for" loop update clause should move the counter in the right direction	Bug	主要
Variables should not be self-assigned	Bug	主要
Non-empty statements should change control flow or have at least one side-effect	Bug	主要
Calls should not be made to non-callable values	Bug	主要
Non-existent operators '=+', '=-' and '=!' should not be used	Bug	主要
"new" operators should be used with functions	Bug	主要
Identical expressions should not be used on both sides of a binary operator	Bug	主要
Array-mutating methods should not be used misleadingly	Bug	主要
Strict equality operators should not be used with dissimilar types	Bug	主要
Setters should not return values	Bug	主要

Comma and logical OR operators should not be used in switch cases	Bug	主要
Collection elements should not be replaced unconditionally	Bug	主要
Bitwise operators should not be used in boolean contexts	Bug	主要
Attempts should not be made to update "const" variables	Bug	主要
Errors should not be created without being thrown	Bug	主要
Collection sizes and array length comparisons should make sense	Bug	主要
"delete" should be used only with object properties	Bug	次要
"with" statements should not be used	Bug	次要
Cross-document messaging domains should be carefully restricted	漏洞	严重
Code should not be dynamically injected and executed	漏洞	严重
Function constructors should not be used	漏洞	严重
Debugger statements should not be used	漏洞	次要
"alert(...)" should not be used	漏洞	次要
Octal values should not be used	坏味道	阻断
Variables should be declared explicitly	坏味道	阻断
"future reserved words" should not be used as identifiers	坏味道	阻断
"switch" statements should not contain non-case labels	坏味道	阻断
Function returns should not be invariant	坏味道	阻断
Switch cases should end with an unconditional "break" statement	坏味道	阻断
Conditionals should start on new lines	坏味道	严重
A conditionally executed single line should be denoted by indentation	坏味道	严重
Equality operators should not be used in "for" loop termination conditions	坏味道	严重
Boolean expressions should not be gratuitous	坏味道	主要
Redundant pairs of parentheses should be removed	坏味道	主要
Functions should not be called both with and without "new"	坏味道	主要
Comma operator should not be used	坏味道	主要
Multiline blocks should be enclosed in curly braces	坏味道	主要
Labels should not be used	坏味道	主要
"switch" statements should not have too many "case" clauses	坏味道	主要
"indexOf" checks should not be for positive numbers	坏味道	主要

Arguments to built-in functions should match documented types	坏味道	主要
Nested blocks of code should not be left empty	坏味道	主要
Dead stores should be removed	坏味道	主要
Array indexes should be numeric	坏味道	主要
Variables and functions should not be redeclared	坏味道	主要
"delete" should not be used on arrays	坏味道	主要
Function parameters with default values should be last	坏味道	主要
Jump statements should not be used unconditionally	坏味道	主要
Two branches in a conditional structure should not have exactly the same implementation	坏味道	主要
Assignments should not be redundant	坏味道	主要
Functions should not be defined inside loops	坏味道	主要
Collection and array contents should be used	坏味道	主要
Default export names and file names should match	坏味道	次要
Boolean checks should not be inverted	坏味道	次要
A "while" loop should be used instead of a "for" loop	坏味道	次要
Function call arguments should not start on new lines	坏味道	次要
Extra semicolons should be removed	坏味道	次要
Return of boolean expressions should not be wrapped into an "if-then-else" statement	坏味道	次要
Unnecessary imports should be removed	坏味道	次要
Wrapper objects should not be used for primitive types	坏味道	次要
Unary operators "+" and "-" should not be used with objects	坏味道	次要
Multiline string literals should not be used	坏味道	次要
"switch" statements should have at least 3 "case" clauses	坏味道	次要
The global "this" object should not be used	坏味道	次要
"catch" clauses should do more than rethrow	坏味道	次要
Unused local variables and functions should be removed	坏味道	次要